5P

УДК 004.272.43

М.К. Раскладкин
НИИ многопроцессорных вычислительных систем им. акад. А.В. Каляева
Южного федерального университета, г. Таганрог, Россия
maxim_work2003@mail.ru

Проблемы синтеза масштабируемых интерфейсов для программируемых логических интегральных схем

В статье рассматриваются проблемы синтеза масштабируемых интерфейсов для реконфигурируемых вычислительных систем на основе программируемых логических интегральных схем. Библиотека типовых масштабируемых интерфейсов позволит специальной системе проектирования автоматически масштабировать, каскадировать и распределять вычислительную структуру задачи, состоящую из вычислительных и интерфейсных блоков, что приводит к сокращению времени разработки прикладных задач.

В НИИ многопроцессорных вычислительных систем ЮФУ для разработки реконфигурируемых вычислительных систем (РВС) широко используются программируемые логические интегральные схемы (ПЛИС).

Реконфигурируемые вычислительные системы на основе ПЛИС обеспечивают более высокую производительность, чем системы на основе стандартных процессоров с «жесткой» неперестраиваемой архитектурой, что достигается путем создания вычислительной структуры, адекватной решаемой задаче [1]. Из недостатков РВС на основе ПЛИС следует отметить значительное время создания таких вычислительных структур.

Любая вычислительная структура на основе ПЛИС имеет в своем составе как вычислительные блоки, выполняющие непосредственно операции над потоками данных в соответствии с заданным алгоритмом, так и интерфейсные блоки, обеспечивающие поступление данных в вычислительные блоки и информационный обмен между ними [2].

Практически каждый год происходит выпуск нового семейства ПЛИС. Происходит переход на новый технологический процесс, что позволяет увеличивать количество логических ячеек на единицу площади кристалла. Растет максимальная тактовая частота, появляются более высокоскоростные интерфейсы по сравнению с кристаллами предыдущих семейств, увеличивается объем встроенной памяти на кристалле, применяются встроенные аппаратные IP-ядра реализации сложных интерфейсов для подключения к современным шинам типа PCI Express, сетевым средствам Gigabit Ethernet, памяти DDR3 и т.д. С выпуском каждой новой серии ПЛИС наблюдается снижение потребляемой мощности и удельной себестоимости логических ячеек по сравнению с кристаллами предыдущей серии.

С ростом аппаратного ресурса ПЛИС появляется возможность реализовать более производительную вычислительную систему путем увеличения количества вычислительных устройств системы, использования высокоскоростных интерфейсов ввода-вывода информации для обмена между ПЛИС в пределах печатной платы вычислительного модуля, так и современных внешних по отношению к РВС интерфейсов для информационного обмена с внешним миром.

Очевидно, что для достижения максимальной производительности необходимо обеспечить такой темп поступления данных на вычислительные устройства, чтобы не возникало простоя оборудования, так называемого эффекта «узкого горла».

Для решения этой проблемы должны использоваться интерфейсные блоки: межблочные, загрузки-выгрузки и аппаратные.

Межблочные интерфейсные блоки используются для информационного обмена между вычислительными блоками внутри ПЛИС.

Интерфейсные блоки для загрузки-выгрузки необходимы при информационном обмене между ПЛИС и для загрузки конфигурационной информации в ПЛИС.

Аппаратные интерфейсы применяются для обеспечения связи РВС между собой с персональным компьютером или иным внешним устройством и обеспечивают обмен информацией со стандартными шинами, например PCI, VME, Ethernet и др.

Таким образом, все три типа интерфейсов должны обеспечить согласованный темп поступления информации от внешнего устройства и прохождения этой информации как внутри одной ПЛИС, между вычислительными блоками, между ПЛИС в пределах одной платы РВС, так и между всеми вычислительными модулями, входящими в состав РВС, для исключения возможности простоя оборудования.

Очевидно, что количество, тип вычислительных устройств в одной ПЛИС и количество задействованных в решении задачи ПЛИС заранее не определены и различны при реализации различных алгоритмов на РВС. Создание набора интерфейсов, учитывающих все возможные сочетания вычислительных устройств, практически невозможно, т.к. существует неограниченное количество сочетаний параметров интерфейсных блоков, таких как: тип сопрягаемых блоков, разрядность сопрягаемых блоков, частота, скважность и скорость потока данных и др.

Следовательно, возникает необходимость в создании библиотеки универсальных интерфейсных блоков, функциональные и структурные параметры которых должны быть настраиваемые. Применение данных интерфейсных блоков должно обеспечивать максимально возможный темп поступления данных на вычислительные устройства.

Не менее важной проблемой является то, что масштабируемые интерфейсы для РВС на основе ПЛИС должны быть независимы от степени параллелизма прикладных задач.

Проблема масштабирования вычислительных блоков достаточно проста при наличии аппаратного ресурса. Однако проблема масштабирования интерфейсных блоков при произвольном количестве взаимосвязанных кристаллов ПЛИС и произвольном количестве базовых модулей РВС представляет собой нетривиальную задачу. До сих пор не предложены стандартные программные масштабируемые интерфейсы для РВС на основе ПЛИС аналогичных MPI или OpenMP для стандартных компьютерных платформ.

Проблема усугубляется тем, что если для стандартных многопроцессорных вычислительных систем (МВС) на основе группы компьютеров, объединенных каналами связи, вычислительная часть программы и процедуры межузлового обмена разделены, то для РВС на основе ПЛИС интерфейсные и вычислительные блоки неразрывно связаны друг с другом.

Для достижения максимальной производительности вычислительные и интерфейсные блоки должны образовывать единую вычислительную систему. Интерфейсные блоки также могут использоваться для соединения вычислительных блоков, реализующих различные методы распараллеливания, при этом темп передачи данных между вычислительными блоками должен быть равен темпу обработки информации каждого вычислительного блока, что необходимо для реализации принципа структурных вычислений. Таким образом, до настоящего времени интерфейсы являются практически неотделимой частью вычислительных блоков, т.е. каждый раз при реализации различных вычислительных алгоритмов приходится использовать одни и те же вычислительные блоки, но с различными интерфейсами.

Следовательно, такое разделение жизненно необходимо, особенно при использовании специальных систем автоматического масштабирования и каскадирования вычис-

лительных блоков. Необходимо создать методы синтеза масштабируемых интерфейсов, что позволит использовать однократно разработанные вычислительные блоки для решения различных прикладных задач.

Появляется проблема выбора и реализации аппаратных и интерфейсов для информационного обмена между ПЛИС. Классические методы построения интерфейсов, используемые в широко распространенных IBM PC совместимых персональных компьютерах, не пригодны. Основной недостаток архитектуры фон-Неймана состоит в том, что она последовательная, т.е. выполнение команд происходит строго друг за другом. Процессор обрабатывает только один поток данных, который поступает от северного моста на материнской плате.

С точки зрения подключения все внешние устройства подключаются к процессору через мосты. Северный мост используется для подключения графического контроллера и памяти, южный подключен к северному мосту и содержит контроллеры шин ввода-вывода типа ISA, PCI, PCI Express, DMA контроллер и других устройств.

Подключение нескольких вычислительных плат, входящих в состав РВС, к управляющему устройству по топологии типа «точка-точка» с использованием стандартных мостов возможно, но приведет к нескольким проблемам. Во-первых, потребуется реализовать несколько аппаратных контроллеров PCI или PCI Express. Реализация этого контроллера в ПЛИС приведет к значительному снижению доступного вычислительного аппаратного ресурса ПЛИС. Во-вторых, при увеличении количества вычислительных модулей РВС возникает проблема каскадирования мостов и общей производительности передачи данных в подобной системе. В-третьих, возникает проблема параллельной высокоскоростной загрузки данных в вычислительные модули РВС, что требуется для задач, например, цифровой обработки сигналов, где объем данных сопоставим с временем вычислений. Не существует стандартных команд для одновременной загрузки больших массивов данных в различные устройства, имеющие различные адреса.

Приведенные выше факторы более чем справедливы для интерфейсов загрузки-выгрузки между ПЛИС, где их число только на одном вычислительном модуле РВС, создаваемом в НИИ МВС ЮФУ, измеряется десятками.

Таким образом, существующий метод построения интерфейсов абсолютно не эффективен для РВС на основе множества ПЛИС, он приведет к избытку оборудования и не обеспечит сбалансированную скорость обмена информацией при масштабировании РВС.

Необходимо разработать интерфейсы нового типа для различных вычислительных задач, учитывающие их специфику. Эти интерфейсы должны быть достаточно простыми, чтобы не занимать вычислительный ресурс ПЛИС и обеспечивать непосредственное взаимодействие блоков между собой. Для многих реализаций алгоритмов на РВС на основе ПЛИС возможен отказ от адресации каждой ПЛИС.

Таким образом, существует проблема создания библиотеки интерфейсных блоков, решающих вышеперечисленные задачи.

Интерфейсные блоки можно представить тремя типами: межблочные интерфейсы, загрузки-выгрузки и аппаратные интерфейсы.

Все типы интерфейсных блоков должны быть, прежде всего, масштабируемыми и решать задачу переносимости при переходе на другое семейство или смене фирмы-производителя ПЛИС.

Для реализации вышеперечисленных требований предлагается использовать язык описания аппаратуры VHDL (Very high speed integrated circuits Hardware Description Language), который широко поддержан многими современными системами проектирования ПЛИС. Использование языка VHDL для синтеза интерфейсных блоков позволяет настраивать структурные и функциональные параметры блока, такие как параметры разрядности, количество каналов, состава блока и др.

Межблочный интерфейс предназначен для информационного обмена между вычислительными блоками и должен обладать свойством масштабируемости в зависимости от степени распараллеливания вычислительных блоков.

Каждый вычислительный блок подразделяется на два типа: конвейерный и процедурный. В отличие от конвейерных вычислительных блоков, где результат выдается каждый такт, в процедурных блоках задержка результата может составлять более двух тактов.

Таким образом, межблочные интерфейсные блоки должны обеспечивать информационный обмен между следующими вычислительными блоками:
– конвейер-конвейер,
– конвейер-процедура,
– процедура-конвейер,
– процедура-процедура,
– макроконвейер,
– вложенный конвейер.

При условии полного совпадения параметров вычислительных блоков используется прямой интерфейс для непосредственного соединения вычислительных блоков друг с другом.

Конвейерные вычислительные блоки можно описать следующими параметрами:
– w – шириной выдаваемых конвейером данных,
– f – тактовой частотой, на которой работает конвейер,
– s – скважностью данных, в общем случае зависящей от времени.

Процедурные вычислительные блоки описываются следующими параметрами:
– w – шириной выдаваемых процедурой данных,
– f – тактовой частотой, на которой работает процедура,
– τ – количеством тактов работы процедуры.

Входными данными для интерфейсного блока типа *конвейер-конвейер* являются: w_1, f_1, s_1 – параметры первого сопрягаемого конвейерного вычислительного блока; выходными – w_2, f_2, s_2 – параметры второго сопрягаемого конвейерного вычислительного блока. Задача интерфейсного блока сопряжения состоит в обеспечении соединения конвейерных вычислительных блоков при их различных параметрах.

В простейшем случае при совпадении всех параметров конвейерных вычислительных блоков используется прямой интерфейс, иначе происходит определение входной ($v_1 = w_1 \cdot f_1 \cdot s_1$) и выходной ($v_2 = w_2 \cdot f_2 \cdot s_2$) скоростей.

При условии $v_1 \le v_2$ необходимо определить входной ($T_1 = 1/f_1 \cdot s_1$) и выходной ($T_2 = 1/f_2 \cdot s_2$) периоды данных, выдаваемых конвейером.

Если ширина данных $w_1 > w_2$, то используется интерфейс $F_1(T_1,T_2,w_1,w_2)$, в другом случае устанавливается интерфейс $F_2(T_2,T_1,w_1,w_2)$.

Если $v_1 > v_2$ и количество данных M_1, поступающих из первого конвейера, меньше определенной величины N, определяемой наличием аппаратного ресурса, то возможно применить буферное FIFO для согласования скоростей v_1 и v_2, в противном случае при отсутствии аппаратного ресурса необходимо снижать скорость работы первого конвейера. Если аппаратного ресурса достаточно, необходимо использовать коммутатор, описываемый параметром k, равным отношению v_1 и v_2.

Если тактовые частоты конвейеров не равны, необходимо согласовать частоты друг с другом.

Интерфейс $F_1(T_1,T_2,w_1,w_2)$ представляет собой регистр, в который записываются данные, поступающие с первого конвейера с частотой f_1, и мультиплексор, имеющий w_1/w_2 информационных входов, каждый разрядностью w_2, и один выход разрядностью w_2.

Мультиплексор управляется счетчиком, который запускается при записи данных в регистр и при достижении значения w_1/w_2 сбрасывается в исходное состояние. Структурная схема интерфейса $F_1(T_1,T_2,w_1,w_2)$ приведена на рис. 1.

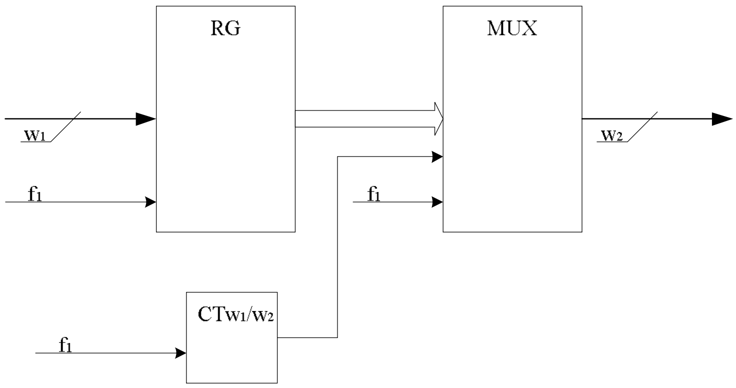


Рисунок 1 – Структурная схема интерфейса $F_1(T_1,T_2,w_1,w_2)$

Интерфейс $F_2(T_2,T_1,w_1,w_2)$ выполняет преобразование, обратное интерфейсу $F_1(T_1,T_2,w_1,w_2)$. Структурная схема интерфейса типа $F_2(T_2,T_1,w_1,w_2)$ приведена на рис. 2.

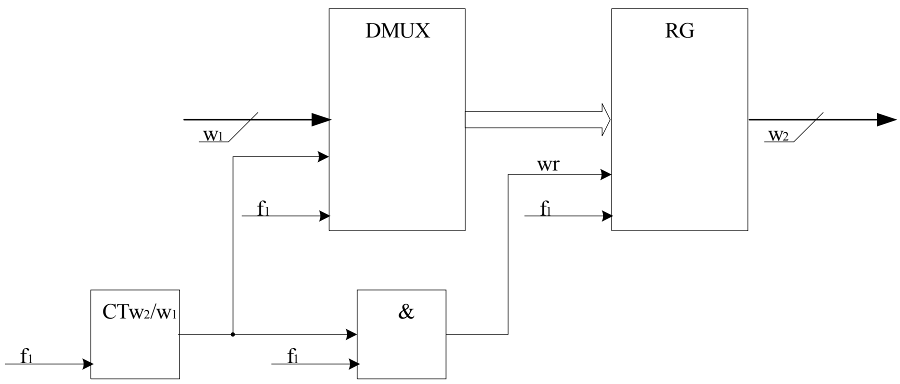


Рисунок 2 – Структурная схема интерфейса $F_2(T_2,T_1,w_1,w_2)$

Данные с разрядностью w_1 поступают в интерфейс $F_2(T_2,T_1,w_1,w_2)$ из первого конвейера с частотой f_1 на демультиплексор. На управляющие входы демультиплексора подается выход счетчика по основанию w_1/w_2. Также выход счетчика поступает на схему сравнения и при достижении значения, равного w_1/w_2, формируется сигнал записи w_r в регистр.

Коммутатор предназначен для коммутирования нескольких конвейерных вычислительных блоков друг с другом. На вход коммутатора поступают вычислительные блоки первого конвейера, на выход – второго, сопрягаемого конвейера. Количество блоков на входе коммутатора и на выходе коммутатора определяется числом k, равным отношению ν_1 и ν_2. Если результат деления ν_1/ν_2 – целое число, то на входе коммутатора имеем один конвейерный вычислительный блок, а на выходе количество блоков равно этому результату. В противном случае на входе коммутатора ν_2 – блоков, на выходе ν_1 – конвейерных вычислительных блоков.

При неравенстве тактовой частоты работы конвейерных вычислительных блоков f_1 и f_2 необходима схема согласования частот, например, реализованная на FIFO.

Входными данными для интерфейсного блока типа *конвейер-процедура* являются: w_1, f_1, s_1 – параметры первого сопрягаемого конвейерного вычислительного блока и w_2, f_2, τ_2 – параметры второго сопрягаемого процедурного вычислительного блока; выход-

ными – w_3, f_3, s_3 – параметры полученного конвейерного вычислительного блока. Задача интерфейсного блока сопряжения состоит в обеспечении соединения конвейерного и процедурного вычислительных блоков при их различных параметрах.

Для реализации данного интерфейса необходимо согласовать разрядности и тактовые частоты конвейерного и процедурного блоков.

Если разрядность данных, выдаваемых конвейером, не соответствует разрядности процедурного блока, необходимо использовать интерфейс $F_1(T_1,T_2,w_1,w_2)$ или $F_2(T_2,T_1,w_1,w_2)$.

Если тактовые частоты конвейера и процедуры не равны, необходимо согласовать частоты друг с другом.

При наличии аппаратного ресурса процедурный блок распараллеливается, и задача сопряжения сводится к распределению потока данных с конвейера на множество процедурных блоков, как показано на рис. 3.

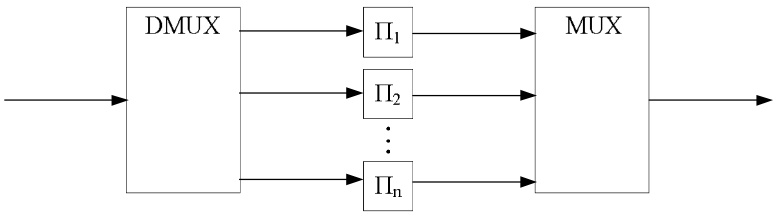


Рисунок 3 – Структурная схема распараллеливания процедурных блоков

Входной поток данных с выхода конвейерного вычислительного блока поступает на демультиплексор, а из него – на процедурные блоки. При количестве процедурных блоков, равном $n = T_{конвейера}/T_{процедуры}$, получим на выходе интерфейса скорость потока данных, равную скорости потока входного конвейерного вычислительного блока. При этом обеспечивается равномерная нагрузка процедурных вычислительных блоков, что ведет к росту удельной производительности системы.

При решении данной задачи зачастую время выполнения процедур достаточно велико, и поэтому не удается масштабировать процедурные блоки с требуемой степенью распараллеливания из-за отсутствия необходимого аппаратного ресурса. В этом случае для построения интерфейса «конвейер-процедура» можно уменьшить разрядность обрабатываемых данных. Это не только приведет к понижению скорости обработки, но и освободит часть аппаратного ресурса конвейера. Этот высвободившийся ресурс можно направить на увеличение степени распараллеливания процедурных блоков.

Входными данными для интерфейсного блока *процедура-конвейер* являются – w_2, f_2, τ_2 – параметры процедурного сопрягаемого блока и w_1, f_1, s_1 – параметры конвейерного вычислительного блока. Выходные данные – w_3, f_2, s_3, – параметры полученного конвейера.

Алгоритм работы интерфейса «процедура-конвейер» аналогичен алгоритму работы интерфейса «конвейер-процедура», за исключением блока распараллеливания процедурного вычислительного блока.

При наличии аппаратного ресурса процедурный блок распараллеливается, и задача сопряжения сводится к распределению потока данных из множества процедурных блоков на конвейер, как показано на рис. 4.

Входными данными для интерфейсного блока типа *процедура-процедура* являются: w_1, f_1, τ_1 – параметры первого процедурного вычислительного блока и w_2, f_2, τ_2 – параметры второго процедурного блока.

При совпадении всех параметров процедурных вычислительных блоков используется прямой интерфейс, иначе происходит определение входной ($\nu_1 = w_1 \cdot f_1 / \tau_1$) и выходной ($\nu_2 = w_2 \cdot f_2 / \tau_2$) скоростей.

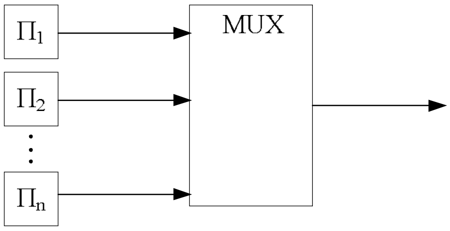


Рисунок 4 – Структурная схема распараллеливания процедурных блоков

При условии $\nu_1 \leq \nu_2$ необходимо определить входной ($T_1 = \tau_1/f_1$) и выходной ($T_2 = \tau_2/f_2$) периоды данных, выдаваемых процедурными блоками.

При ширине данных $w_1 > w_2$ используется интерфейс $F_1(T_1,T_2,w_1,w_2)$, в противном случае устанавливается интерфейс $F_2(T_2,T_1,w_1,w_2)$.

Если $\nu_1 > \nu_2$ и количество данных M_1, поступающих из первой процедуры, меньше определенной величины N, определяемой наличием аппаратного ресурса, то возможно применить буферное FIFO для согласования скоростей ν_1 и ν_2.

В противном случае при отсутствии аппаратного ресурса необходимо снижать скорость работы первой процедуры. Если аппаратного ресурса достаточно, необходимо использовать коммутатор, описываемый параметром k, равный отношению ν_1 и ν_2. Коммутатор аналогичен коммутатору, описанному для интерфейса типа конвейер-конвейер, за исключением того, что на его входы подаются вычислительные блоки первой процедуры, а на выход – второй сопрягаемой процедуры.

Интерфейс сопряжения *макроконвейер* предназначен для соединения конвейерных и процедурных вычислительных блоков и выполняет функции, аналогичные интерфейсу конвейер-процедура.

Если в интерфейсе конвейер-процедура разрядность обрабатываемых данных и количество распараллеленных процедур достаточно велико, то аппаратная реализация мультиплексора и демультиплексора затруднительна. В таком случае целесообразно использовать интерфейс сопряжения макроконвейер, структурная схема которого показана на рис. 5.

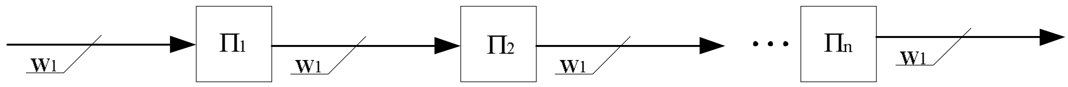


Рисунок 5 – Структурная схема интерфейса макроконвейер

Реализация данного интерфейса предусматривает установку распараллеленных процедурных блоков последовательно друг за другом. В таком случае каждый процедурный блок $П_i$ содержит интерфейс для передачи данных, который содержит один мультиплексор с двумя входами разрядностью w_1 и одним выходом такой же разрядности (рис. 6).

Первое данное поступает с выхода конвейерного вычислительного блока на процедуру $П_1$, второе поступает через мультиплексор процедуры $П_1$ на вход процедуры $П_2$ и так далее до заполнения макроконвейера. Результат работы: каждая процедура выгружает в последовательную шину. При количестве процедурных блоков, равном $n = T_{конвейера}/T_{процедуры}$, получим на выходе интерфейса скорость потока данных, равную скорости потока входного конвейерного вычислительного блока.

Интерфейс *вложенный конвейер* предназначен для сопряжения нескольких информационно независимых конвейерных вычислительных блоков с одним конвейерным вычислительным блоком при условии, что скважность потока данных s сопрягаемого конвейера много больше скважности любого из входных сопрягаемых конвейеров. Входные конвейеры должны быть информационно независимы.

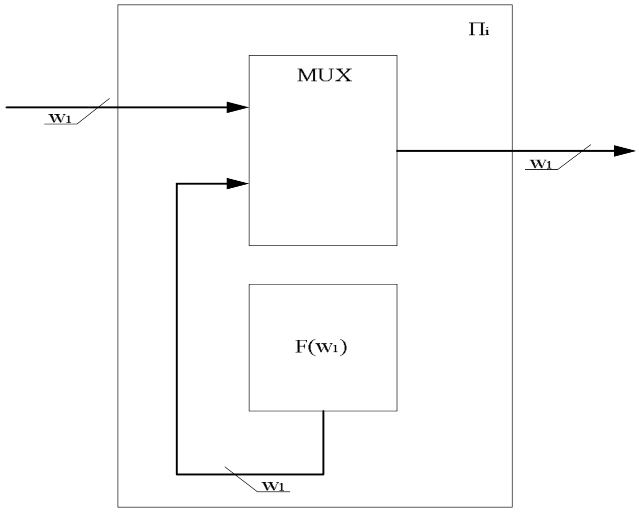


Рисунок 6 – Структурная схема интерфейса процедуры

Входными данными для интерфейсного блока «вложенный конвейер» являются $w1_i$, $f1_i$, $s1_i$, $i = 2..n$ – параметры входных сопрягаемых конвейерных вычислительных блоков, выходными – w2, f2, s2 – параметры второго сопрягаемого конвейерного вычислительного блока.

Прежде всего, необходимо вычислить скорости всех входных конвейерных блоков. При несовпадении данных скоростей необходимо с помощью интерфейса конвейер-конвейер привести каждый входной конвейер к конвейеру с минимальной скоростью.

Если ширина данных приведенных входных конвейеров отличается от ширины данных выходного конвейера, необходимо обеспечить согласования с помощью интерфейса $F1(T1\nu_{min},T2,w1\nu_{min},w2)$ или $F2(T2,T1\nu_{min},w1\nu_{min},w2)$, аналогичных $F_1(T_1,T_2,w_1,w_2)$ и $F_2(T_2,T_1,w_1,w_2)$, описанных для интерфейса конвейер-конвейер. При неравенстве тактовых частот конвейеров необходимо их согласовать.

Структурная схема распределения потоков данных входных конвейеров на выходной конвейерный блок с помощью коммутатора показана на рис. 7.

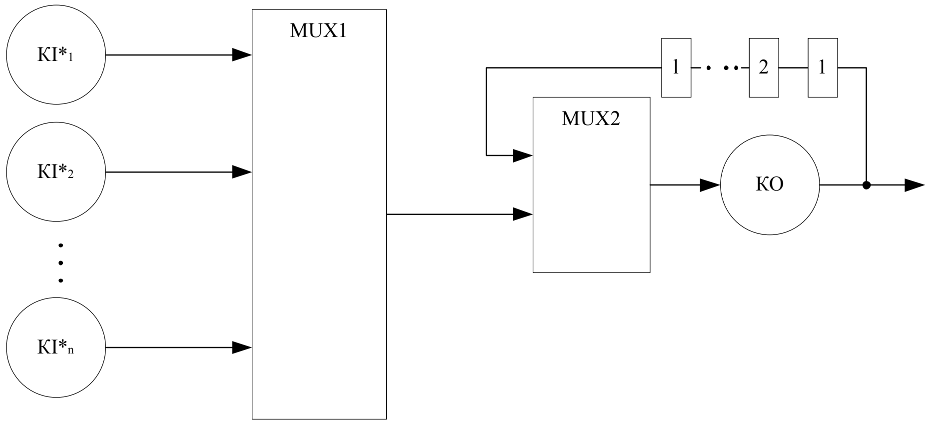


Рисунок 7 – Структурная схема распределения потоков данных с использованием коммутатора

Если коммутатор последовательно подает по одному данному каждого входного конвейера на выход, перебирая входные конвейеры, формируя входную последовательность:

$$KI_1^1, KI_1^2 ... KI_1^m, KI_2^1, KI_2^2 ... KI_2^m ... KI_n^1, KI_n^2 ... KI_n^m,$$

где m – количество данных, выдаваемых входными конвейерами, то на выходе интерфейса получим последовательность данных со скважностью, равной скважности конвейерного блока КО.

Коммутатор может выдавать данные на вход конвейера КО порциями, от разных входных конвейеров, с количеством данных, не превышающим параметр l – число данных циркулирующих в обратной связи выходного конвейера, очевидно, что это число не может превышать глубину конвейера. В таком случае, при $l = n$, коммутатор будет формировать следующую входную последовательность:

$$KI_1^1, KI_2^1 \ldots KI_n^1, KI_1^2, KI_2^2 \ldots KI_n^2 \ldots KI_1^m, KI_2^m \ldots KI_n^m,$$

и на выходе интерфейса получим последовательность данных, следующих каждый такт.

При $n > l$ входная последовательность примет вид:

$$KI_1^1, KI_2^1 \ldots KI_l^1, KI_1^2, KI_2^2 \ldots KI_l^2 \ldots KI_{l+1}^m, KI_{l+2}^m \ldots KI_{2l}^m \ldots$$
$$\ldots KI_{l+1}^1, KI_{l+2}^1 \ldots KI_{2l}^1, KI_{l+1}^2, KI_{l+2}^2 \ldots KI_{2l}^2 \ldots KI_{l+1}^m, KI_{l+2}^m \ldots KI_{2l}^m \ldots$$
$$\ldots KI_{n-l}^m, KI_{n-l-1}^m \ldots KI_n^m,$$

и на выходе интерфейса также получим последовательность данных, следующих каждый такт.

Очевидно, что, при условии $n < l$, на выходе интерфейса получим последовательность выходных данных со скважностью l/n.

Таким образом, предложенная библиотека масштабируемых интерфейсов для реконфигурируемых вычислительных систем на основе ПЛИС позволит объединить в единой вычислительной структуре устройства, реализующие вычисления различными методами параллельно-конвейерной обработки, что значительно упростит создание таких структур и приведет к сокращению времени разработки прикладных задач.

Использование данной библиотеки как элемента специальной среды разработки масштабируемых компонентов вычислительных структур для РВС позволит автоматически масштабировать и распределять аппаратную реализацию алгоритма по вычислительным устройствам РВС.

Литература

1. Каляев А.В. Модульно-наращиваемые многопроцессорные системы со структурно-процедурной организацией вычислений / А.В. Каляев, И.И. Левин. – М. : Изд-во «Янус-К», 2003. – 380 с.
2. Каляев И.А. Реконфигурируемые мультиконвейерные вычислительные структуры / И.А. Каляев, И.И. Левин, Е.А. Семерников, В.И. Шмойлов ; под общ. ред. И.А. Каляева. – Ростов/Д. : Изд-во ЮНЦ РАН, 2008. – 320 с.

М.К. Раскладкін

Проблеми синтезу масштабованих інтерфейсів для програмованих логічних інтегральних схем

У статті розглядаються проблеми синтезу масштабованих інтерфейсів для реконфігурованих обчислювальних систем на основі програмованих логічних інтегральних схем. Бібліотека типових масштабованих інтерфейсів дозволить спеціальній системі проектування автоматично масштабувати, каскадувати і розподіляти обчислювальну структуру задачі, яка складається з обчислювальних та інтерфейсних блоків, що приводить до скорочення часу розробки прикладних задач.

M.K. Raskladkin

Problems of Synthesis of Scalable Interfaces for Programmable Logic Integrated Circuits

Problems of synthesis of scalable interfaces for reconfigurable computer systems on the basis of programmable logic integrated circuits are discussed in the paper. Computational structure of the task, which consists of computing and interface blocks, may be automatically scaled, cascaded and distributed using library of standard scalable interfaces and special development system. Due to this time of applied tasks development will be reduced.

Статья поступила в редакцию 29.07.2009.